ERDC/CERL TR-12-21

US Army Corps
of Engineers®
Engineer Research and
Development Center

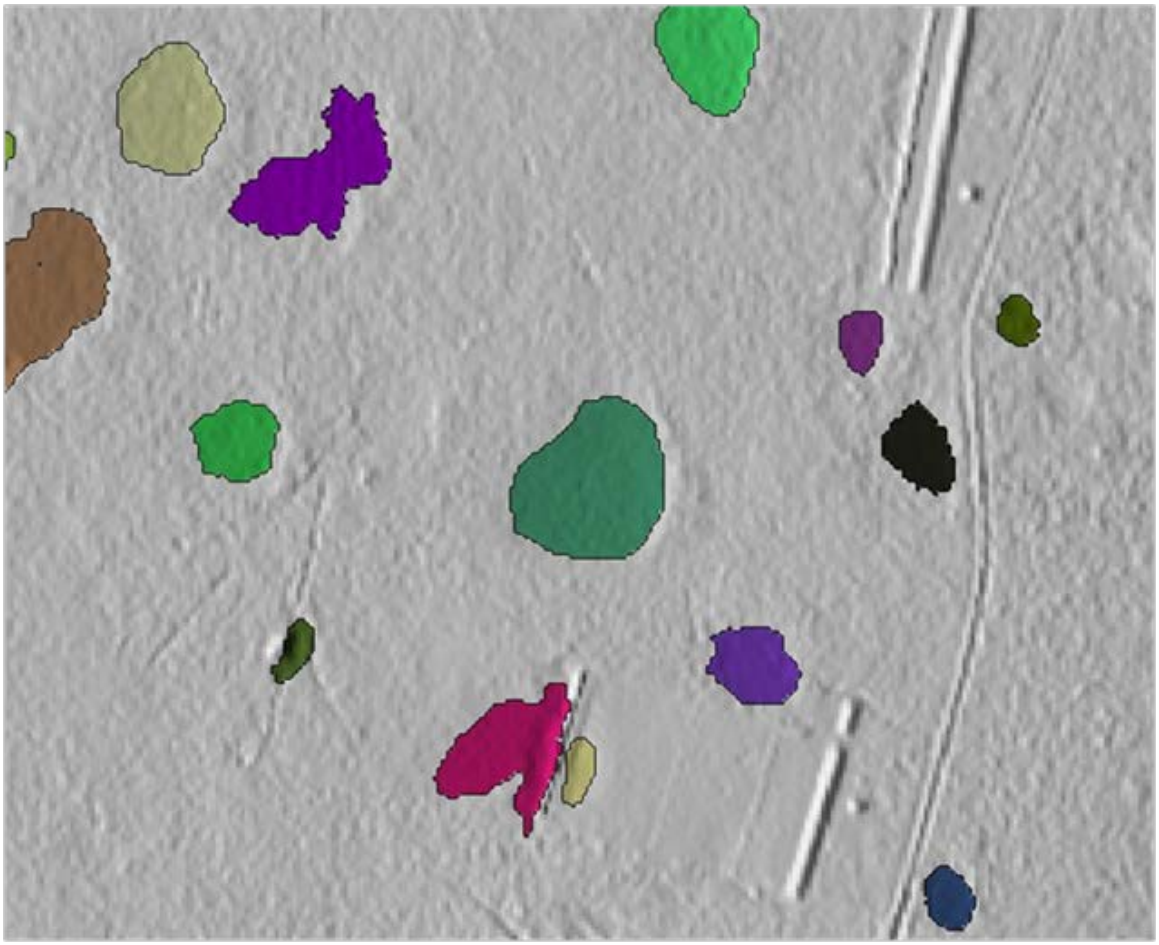
ERDC
INNOVATIVE SOLUTIONS
for a safer, better world

Center Directed Research Program

Digital Discover of Ephemeral Ponds

James D. Westervelt

August 2012



Engineer Research and
Development Center

The US Army Engineer Research and Development Center (ERDC) solves the nation's toughest engineering and environmental challenges. ERDC develops innovative solutions in civil and military engineering, geospatial sciences, water resources, and environmental sciences for the Army, the Department of Defense, civilian agencies, and our nation's public good. Find out more at www.erdc.usace.army.mil.

To search for other technical reports published by ERDC, visit the ERDC online library at http://acwc.sdp.sirsi.net/client/default.

ERDC/CERL TR-12-21
August 2012

Digital Discover of Ephemeral Ponds

James D. Westervelt

Construction Engineering Research Laboratory (CERL)
US Army Engineer Research and Development Center
2902 Newmark Dr.
Champaign, IL 61822-1076

Final Report

Prepared for Headquarters, US Army Corps of Engineers
Washington, DC 20314-1000

Under Work Unit 33143

Abstract

The US Endangered Species Act requires Federal agency land owners, including military installations, to manage their lands in a manner that enhances the survival of Federally listed species. Many species become at risk due to the loss of "ephemeral ponds," depressions in the landscape that occasionally become ponds after sufficient rainfall. This report developed a modeling approach using the Geographic Resources Analysis Support System Geographic Information System (GRASS GIS) software, augmented with a NetLogo-based model to add behavior to those ponds within the NetLogo spatially explicit simulation-modeling environment. This tool will allow installation land managers to quickly and accurately locate and measure ephemeral ponds to support species that rely on that environment for breeding.

Table of Contents

List of Figures and Tables

Figures

Tables

Preface

This study was conducted for the Engineer Research and Development Center (ERDC) Director under the Center Directed Research Program through the project "Integrated Modeling and Risk Analysis for the Environmental Consequences of Climate Change." The ERDC technical monitor was Dr. Todd Bridges, CEERD-EM-D.

The work was performed by the Ecological Processes Branch (CN-N) of the Installations Division (CF), US Army Engineer Research and Development Center, Construction Engineering Research Laboratory (ERDC-CERL). Special appreciation is owed to Fort Stewart, GA personnel John Macey, Larry Carlile, and Ron Owens, who generously provided vector-based maps and spreadsheets pertaining to ponds on that installation. At the time of publication, William D. Meyer was Chief, CEERD-CN-N; Dr. John T. Bandy was Chief, CEERD-CN; and Dr. Alan B. Anderson was the Technical Director for Military Ranges and Lands. The Deputy Director of ERDC-CERL was Dr. Kirankumar Topudurti and the Director was Dr. Ilker Adiguzel.

CERL is an element of the US Army ERDC, US Army Corps of Engineers. The Commander and Executive Director of ERDC is COL Kevin J. Wilson, and the Director of ERDC is Dr. Jeffery P. Holland.

1 Introduction

1.1 Background

The US Endangered Species Act (ESA) (USFWS 1973) requires Federal agency land owners, including military installations, to manage their lands in a manner that enhances the survival of Federally listed species. Listed species can be "at risk," "endangered," or "threatened" because of the loss of one or more environmental factors. Reasons for such environmental losses can include loss of habitat, fragmentation of habitat, competition from invasive species, loss of natural fire regimes, increased hunting, accidental killing by road vehicles, increased predation, and new diseases. Many species become at risk due to the loss of "ephemeral ponds," depressions in the landscape that occasionally become ponds after sufficient rainfall. Ephemeral ponds are common in areas where the landscape is nearly flat, where there is a relatively impermeable clay layer beneath the soil, and where seasonal rains provide enough water to leave water standing in the depressions.

Ephemeral ponds are often perceived as a nuisance to humans and are typically drained or filled to support agriculture, neighborhoods, and commercial areas. However, species in the competitive natural world can exploit such ponds as breeding areas that are not expected or occupied by potential predators. Adults can bide their time in surrounding areas for one or more years and then begin breeding when the conditions are just right. Because the area is dry most of the time, aquatic predators are minimized, and when the ponds fill, terrestrial predators avoid the water, resulting in increased breeding success. This report describes a general approach for modeling ephemeral ponds, which installation land managers can use to support species that rely on that environment for breeding. (Historically, this was possible only through expensive on-the-ground elevation acquisitions or through intensive human 3-D digitization of elevation contours using images that form a stereo pair.)

1.2 Objectives

The objectives of this work were to develop a modeling approach to quickly and accurately locate and measure ephemeral ponds using the Geographic Resources Analysis Support System Geographic Information Sys-

tem (GRASS GIS) software, and then to add behavior to those ponds within the NetLogo spatially explicit simulation-modeling environment.

1.3 Approach

This work used standard Light Detection and Ranging (LIDAR) data (for the Fort Stewart, GA landscape), which were processed into vector maps of ponds, to model hydrology within a spatially explicit simulation model. An ERDC-developed geographic system and an open-source spatially explicit simulation-modeling environment are used to generically describe and demonstrate the process the steps.

Appendix A to this report contains code used to turn LIDAR data into a 5m resolution Digital Elevation Model (DEM). Appendix B contains NetLogo code used to establish and simulate ephemeral pond filling and emptying in response to historic weather.

1.4 Mode of technology transfer

It is anticipated that the results of this work will be used in current research undertaken to model and forecast historic population sizes of Federally listed species, specifically, that of the Flatwood Salamander at Fort Stewart, GA.

This report will be made accessible through the World Wide Web (WWW) at URLs:

http://www.cecer.army.mil

http://libweb.erdc.usace.army.mil

2 Tools

The procedures described in this report are demonstrated using the GRASS and the NetLogo spatially explicit simulation-modeling package. GRASS (Neteler and Mitasova, 2007) was originally developed at the Army Corps of Engineer's Construction Engineering Research Laboratory (Westervelt et al. 1992). GRASS is a suite of GIS map analysis and display programs that have been developed by dozens of software engineers at many research laboratories across the world. Capabilities within GRASS support the import, export, analysis, and display of raster, vector (two and three dimensional [2-D and 3-D]), image, and LIDAR datasets. Readers can begin their exploration of the GRASS software at: http://grass.osgeo.org/

NetLogo (Wilensky 1999) is a user-friendly agent-based modeling environment created by Dr. Uri Wilensky. NetLogo development continues; new releases are regularly published. It runs on most desktop computer platforms, including Microsoft Windows®, Apple operating system OS X, and Linux. The language actually used to develop simulation models is a "dialect" of Logo that has been implemented in nearly 100 different systems. NetLogo differs from these in many respects, particularly in its ease of use and greater power. Interested readers can find details about this in the NetLogo frequently asked questions (FAQ) document and the Programming Guide, both of which are available on the NetLogo web site at: http://ccl.northwestern.edu/netlogo/

NetLogo has been adopted for many ERDC-developed spatially explicit simulation models. Westervelt and Cohen (2012) document models of river nutrients, endangered species, human social interactions, and species interactions across landscapes. This book also presents the approach to multidisciplinary spatially explicit modeling that informed the development of these models. Railsback and Grimm (2011) provide an excellent general (textbook) introduction to modeling with NetLogo.

3 Finding Ponds

The key to finding ephemeral ponds is to acquire or develop a good DEM, a gridded array of landscape elevations over a study area. The resolution required is based on local needs. In some applications, a resolution of tens of meters may be adequate, while in other areas a much finer resolution may be required. DEMs are now commonly available at resolutions of 30m and 100m, which is often not adequate. A growing number of areas are now covered with LIDAR data, which can be processed into DEMs at most any resolution.

3.1 Acquire LIDAR data

LIDAR data are acquired via an aircraft flying over a study area. Laser beams are rapidly shot into the landscape during flight and the light is reflected back to the aircraft where it is analyzed. On-board software uses precise information about the aircraft's location in space; its roll, pitch, and yaw; and the 3-D angle at which the laser is shot, along with the distance to the reflection to precisely calculate the location in 3-D space where the laser reflected from some surface. The result is a myriad of 3-D coordinates which, when displayed, can appear like a cloud of particles floating in space (Figure 1). Those points may be associated with the ground, with manmade structures or items (e.g., vehicles), or with vegetation. That vegetation may be the tops of trees, the leaves of bushes, or grass on the ground. Water does not reflect the laser, and items that are highly reflective will generally not reflect back to the aircraft; either case results in voids in the resulting dataset.

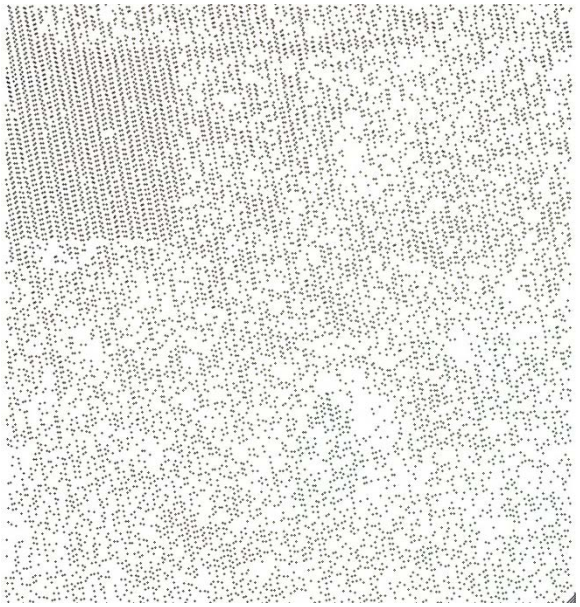


Figure 1. Sample LIDAR dataset.

3.2 Process LIDAR data into a DEM

Processing raw LIDAR data into a DEM, requires the following major steps:

1. Select a subset of the data that represent hits on the ground.
2. Create a mathematical representation of a surface that intersects the ground points.
3. Identify the height of that surface at regular intervals, which represent the center of raster GIS grid cells.

GRASS software offers a substantial number of LIDAR data processing tools to complete these tasks. The raw data reside in an ASCII file that contains 3-D x,y,z coordinates. A sampling of the data file where the columns are x-coordinate, y-coordinate, (in this case) Universal Transverse Mercator coordinate system (UTM) easting and northing values for Zone 17, and elevation (expressed here in feet):

```
426898.57,3554041.57,23.29
426897.92,3554034.23,23.35
426897.71,3554031.85,23.29
426897.28,3554026.98,23.18
426897.06,3554024.55,22.99
426896.86,3554022.26,23.05
```

To select a subset of the data, one must first establish the region of interest and the desired resolution. The extent of the dataset can be determined with the command:

```
r.in.xyz -s fs=, in=all.xyz out=test
```

The –s request reporting of the extent of the data found in the all.xyz file; "fs=," indicates that the field separator in the file is a comma. For the Fort Stewart dataset, the command results in this report:

Range:	min	max
x:	415800.00	474999.99
y:	3523600.00	3556297.15
z:	-9.64	58.89

The minimum bounding box for this area can be set with this command, which also sets the grid resolution to 5m:

```
g.region w=415800 e=475000 s=3523600 n=3556300 res=5
```

Actually, any region within this bounding box can be chosen. This particular region consists of 77,433,600 cells. Now, the r.in.xyz program is used to identify the total number of points associated with each grid cell and the elevation of the lowest point:

```
r.in.xyz fs=, in=all.xyz out=min method=min
r.in.xyz fs=, in=all.xyz out=n method=n
```

The option "method=min" selects the minimum z coordinate found within each cell and "method=n" counts the total number of coordinates associated with each cell. The more complex the vegetation on the landscape, the more hits are needed to provide the confidence that at least one of the points is associated with the ground. One can filter out the cell minimum values if there are an insufficient number of hits in the cell. For example, the command that deletes values that are associated with cells that have fewer than five hits is:

```
r.mapcalc 'min=if(n<3,null(),min)'
```

This filter looks at the value of each patch in the map named "n," created above, and if that value is less than 5 (i.e., 4 or fewer LIDAR points in the patch), sets the value of the patch to a null value; otherwise it retains the original value. This step may not be necessary if the xyz data are already a selected subset of the raw LIDAR data that represent the ground.

At this point, the raster DEM is likely to have a lot of "holes," gridcells assigned a null value because of no LIDAR hits. There are several approaches for assigning reasonable values to those locations. The inverse distance weighting (IDW) approach looks at the areas with values surrounding each missing data point and averages the surrounding values, with each adjusted in importance based on its distance. That is values associated with neighboring cells are weighted as more important than values in more distant cells. The regularized spline with tension (RST) and bicubic or bilinear spline (bspline) approaches generate a 2-D equation based on the data surrounding each missing data that describe a continuous curved surface. This approach can generate higher and lower values than surrounding areas where there appears to be peak or valley at the missing data locations.

To use the spline approaches, the raster file "min" must be converted to a vector point map as follows:

```
r.to.vect -z feature=point in=min out=min
```

Then, the vector point file can be processed with the RST process to generate a continuous raster map:

```
v.surf.rst layer=0 in=min elev=dem
```

These two previous commands require a substantial amount of computer memory. For example the r.to.vect command running on the Fort Stewart LIDAR data require about 17 Gbytes of memory and the v.surf.rst program requires far more memory than is available on most available machines. In addition, these commands are single-threaded and do not make use of multiple core computers. Therefore, an alternate approach was developed in the form of this shell script (included in Appendix A to this report), which contains explanatory comments.

Two more steps are required. First, each of the resulting DEM maps needs to be trimmed by the extra 25m and then the resulting maps need to be combined together into one DEM for the installation. The r.mapcalc program can be used in both steps.

This completes the transition of a file containing the x, y, z coordinates of LIDAR data into a 5m DEM. The LIDAR points (Figure 1) become the DEM (Figure 2).

3.3 Find ponds and create the pond vector file

3.3.1 Find depressions

The DEM created above is likely to have depressions, some of which are likely to be associated with ephemeral ponds (Figure 3). Ideally, the LIDAR was acquired during a dry period that has left the ponds empty. Laser shots that strike water result in no data and, while a “no data” result can indicate water, it is not useful for identifying the bottom structure (floor) of a water body.

Assuming that the LIDAR was acquired when ponds are empty, the next step is to run a watershed structure analysis program designed to find watersheds and streams. The r.terraflow command processes the LIDAR-based elevation file “lidar_elev” to create a suite of derivative files. This program, like others, begins by finding areas that do not drain, and “fills” those areas to a level that allows simulated rain to flow across them. This filled version of the DEM can be saved.

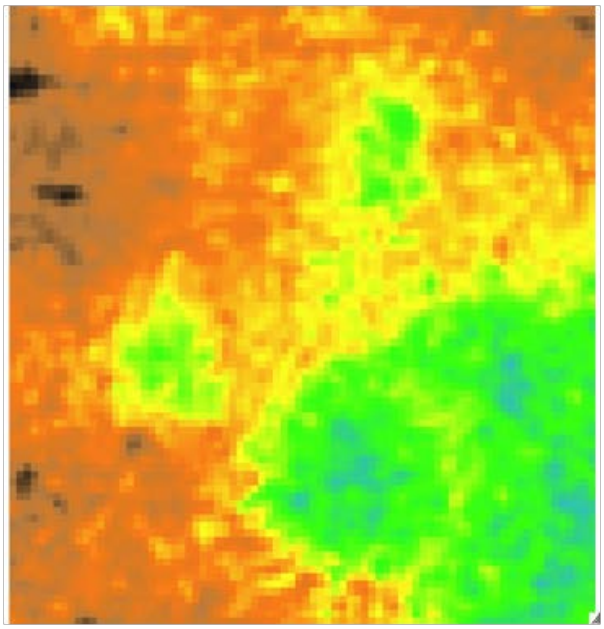


Figure 2. Sample DEM.

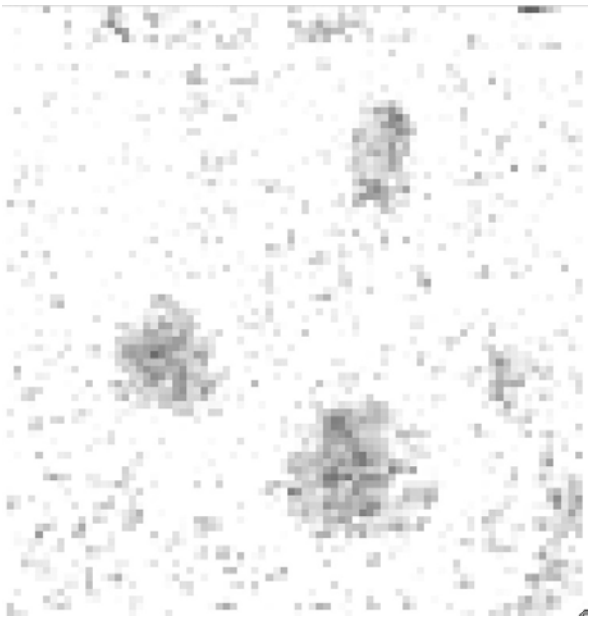


Figure 3. Sample pond depth.

The command below saves that file as "elev_filled." By subtracting the original elevation at every location from its filled elevation, the depth of the water when the area is filled can be calculated using the simple r.mapcalc formula :

```
r.terraflow lidar_elev filled=elev_filled \
        accum=elev_accum memory=2000 \
        dir=elev_dir swater=elev_sink tci=elev_tci
r.mapcalc pond_depth='elev_filled – lidar_elev'
```

3.3.2 Neighborhood averaging to reduce the influence of ground vegetation

A challenge with LIDAR data is dealing with dense clumps of ground vegetation. This can show up in the above calculated pond_depth map. One way to deal with this is to simply average the depth over a range of cells. In the r.neighbors calculation, every location is given a new pond depth as the average of the pond depth of a circular area that is 7 cells in diameter:

```
r.neighbors pond_depth out=x1 size=7 -c --o
```

Experimentation is required in each analysis to determine whether or not to use this step and, if so, what the range for averaging should be.

3.3.3 Find cells that have ponds deeper than 10 cm

The result from the last calculation can show a dense array of depressions in the data. To begin filtering through this data to identify likely ponds, one selects areas that have greater than a desired depth. The following example uses 0.1 m (10 cm) as the depth threshold:

```
r.mapcalc x2='if(x1 >= .1, 1, null())'
```

This results in a binary map showing all grid cells that appear to be associated with depths greater than or equal to 10 cm (Figure 4).

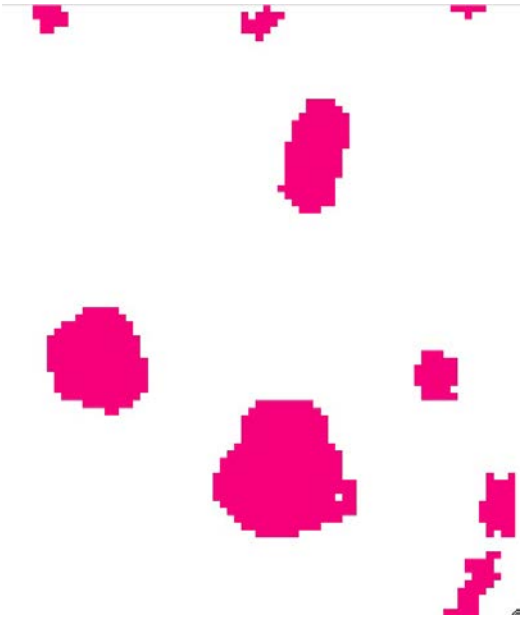


Figure 4. Sample ponds.

3.3.4 Give ponds a unique value and select those within size range

At this point, it is desirable to begin uniquely identifying ponds as sets of contiguous locations that can hold water at (or above) the minimum depth. The r.clump performs this step and turns the binary pond map (x2) into a map giving each set of contiguous cells unique values. The second command blends three commands together to select only those ponds that meet certain size requirements:

```
r.clump x2 out=x3 --o
r.stats -c x3 | awk 'BEGIN { a=1} { if ($2 > 50 && $2 < 4000) {printf ("%d =
        %d\n," $1, a); a = a + 1 }}' | r.reclass --o x3 out=x4
```

The "r.stats –c" command results in a count of cells associated with each category (pond number) as a table where column 1 is the pond number and column 2 is the cell count. This is piped (using the "|" symbol) into an awk program that essentially reformats the table into a form that allows the "r.reclass" program to select specific ponds. The ponds selected are those that are associated with more than 50 cells (0.125 Ha) and less than 4000 cells (10 Ha). The idea is that smaller ponds are perhaps puddles that need not be modeled and larger ponds are lakes or portions of streams or rivers. Again, different thresholds may be more appropriate for other areas.

3.3.5 Create vector file of ponds

At this point, the x4 map, when displayed, shows a set of ponds that meet the various selection criteria. This map can be used directly in a simulation model if that model works with raster files. All grid cells coded with the same number are guaranteed to be contiguous and part of an area within the desired thresholds. This can now be recast in the form of a vector map using the r.to.vect program. Here, cell corners are smoothed using the "-s" argument and the contiguous cells are treated as a single area in the output file, here called "ponds":

```
r.to.vect x4 -s out=ponds
        feature=area --o
```

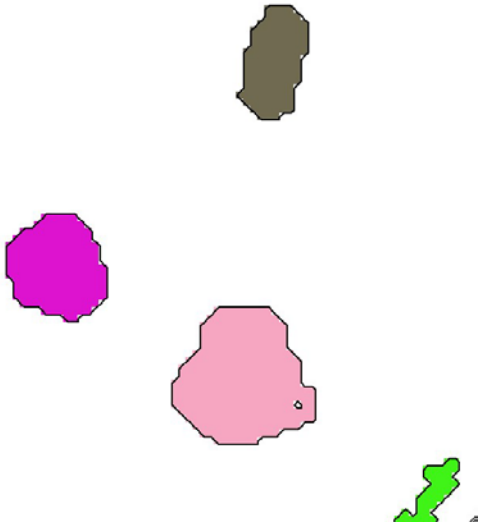


Figure 5. Selected ponds.

Figure 5 shows the selected ponds (x4) and the associated vector files.

3.4 Add characteristics to the pond file

Vector files are associated with tabular information that defines characteristics associated with each vector entity. The commands described below calculate information based on the raster version of the ponds and then adds that information to the vector map's table.

3.4.1 Find size of each pond and add to the table

The first step is to calculate the size of each pond, which is accomplished with the r.stats program using the "-c" argument, which asks for counts of cells that have the same value:

```
r.stats -c x4 | sort -n > /tmp/pond_size
```

The result is sorted and stored temporarily in a file called "pond_size" in the "/tmp" directory.

To put that information into the vector table, a new column is created in the table using the following command:

```
echo 'ALTER TABLE ponds ADD COLUMN area_ha double' | \
    db.execute
```

With the new column created, information can be added based on the information temporarily stored in the "pond_size" file. This information is processed with a simple awk program that converts the cell count values into a hectare value and then adds text that turns each record into a command that instructs the "db.execute" to add the information in the table where it is associated with the correct pond:

```
cat /tmp/pond_size | \
    awk '{printf("UPDATE ponds SET area_ha=%f WHERE value=%d;\n," $2 *
    0.0025, $1)}' | \
    db.execute
```

3.4.2 Find maximum depth for each pond and add to the table

The maximum depth is found by using the r.stats program to generate a list of all pond cells tabulating their pond number and associated pond depth:

```
r.stats -n -c -1 x4,lidar_pond_depth | sort -n | \
    awk 'BEGIN{p=0;d=0;} \
    {if ($1 != p) { if ( d != 0) { printf("%d %f\n," p, d)} p = $1; d = $2} \
    else \
    { if ($2 > d) d = $2}} \
    END{ printf("%d %f\n," p, d)}' \
    > /tmp/pond_depth
```

The sort program ensures that records for all ponds are grouped together and the awk program searches to find the greatest depth and outputs that value with the pond value, storing the result in a temporary file, "pond_depth."

As for the pond size, a new column is created to store the information for the ponds vector file and the information in the "pond_depth" file is edited to provide instructions to db.execute to place the information appropriately in the pond vector map table:

```
echo 'ALTER TABLE ponds ADD COLUMN depth double' | \
    db.execute
cat /tmp/pond_depth | \
    awk '{printf("UPDATE ponds SET depth=%f WHERE value=%d;\n," $2,
    $1)}' | \
    db.execute
```

3.4.3 Find watershed size for each pond

One of the outputs of the r.terraflow operation is an accumulation map. For each grid cell, this contains the total area that drains through the cell, i.e., the size of the watershed for every location. The next step is to discover the size of the watershed that drains into each pond. Logically, this is the largest watershed size of all the locations on the edge of the pond. The associated cell is the pond's spillover location.

This process begins by identifying all of the cells that border each pond, starting with the raster pond map, "x4," which is grown out one cell with the r.grow program. Then, just those grown areas are identified with r.mapcalc:

```
r.grow x4 out=x5 –o
r.mapcalc x5='if(x5 > 0 && isnull(x4), x5, null())'
```

Using the r.stats program, a table of all of the pond edge cells that contains the pond number and the watershed size is generated:

```
r.stats -n -c -1 x5,elev_accum | sort -n | \
        awk 'BEGIN{p=0;n=0;} {if ($1 != p) { if ( n != 0) { printf("%d %f\n," p, n)} p
        = $1; n = $2} else { n = n + $2}} END{ printf("%d %f\n," p, n)}' \
        > /tmp/watershed_size
```

This table is piped into an awk program that selects the record for each pond with the biggest watershed size, and is stored in a temporary file.

3.4.4 Add watershed size to pond vector file

As with the other information added to the vector map, a new column is created and the information in the "watershed_size" temporary file is formatted into instructions for db.execute, which updates the vector data as desired. Note that the awk program converts cell counts to hectares:

```
echo 'ALTER TABLE ponds ADD COLUMN accum_ha double' | \
        db.execute
cat /tmp/watershed_size | \
        awk '{printf("UPDATE ponds SET accum_ha=%f WHERE value=%d;\n," $2
        * 0.0025, $1)}' | \
        db.execute
```

3.5 Summary

This chapter has described the logic that converts raw LIDAR data (x,y,z format) into a vector file of individual ephemeral ponds with associated

pond depth, pond size, and the size of the watershed feeding the pond. The primary steps were:

1. Acquire LIDAR data.
2. Create a DEM:
   a. Select data that represent an area of interest.
   b. Select points that likely represent laser strikes on the ground.
   c. Create a mathematical 2-D model of the surface suggested by the data points.
   d. Query that model (Figure 6) to create a raster-based DEM.
3. Locate ponds in the DEM:
   a. Find depressions in the DEM.
   b. Uniquely label each depression.
   c. Convert the raster-based ponds to vector maps.
4. Calculate the maximum and average depths for each pond.
5. Calculate the size of the watershed feeding each pond.
6. Attach pond depths and watershed sizes to the vector version of each pond.

Sample GRASS and Unix commands were provided to guide the processing of LIDAR data.

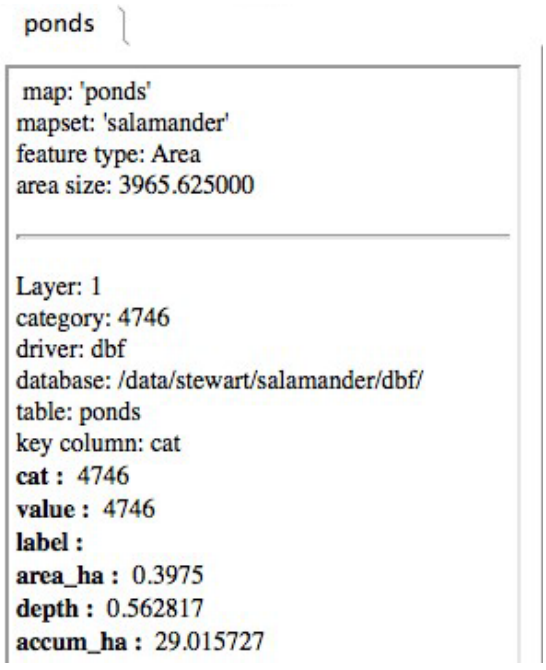


Figure 6. Pond query.

4 Modeling with Ephemeral Ponds

This chapter describes an application of LIDAR-derived ephemeral ponds on Fort Stewart, GA. The objective of this model is to estimate the depth of water in the ponds over time in response to rainfall. The approach is very simple; each pond is treated as a leaky bucket that is fed by a watershed that is capable of absorbing a certain amount of rain each week. The "leakiness" is associated with water loss due to percolation into the groundwater, evaporation, and vegetation-mediated transpiration.

4.1 State variables and scales

The model employs a weekly time step. Ponds, established from LIDAR-based digital elevation at a resolution of 5m, behave hydrologically.

4.2 Process overview and scheduling

Ponds are modeled as agents defined by location, size, watershed area, and depth. Over time, the model tracks water depth in each pond in response to rainfall. At each weekly time step, any water in a pond is drained by a fixed depth change. Rain is added to the model based on an input file containing historic (or fictional) rain data associated with the simulation time frame. The watershed absorbs some of the rain, and the remaining water is added to the associated ponds. Ponds accumulate water and overflow once filled.

4.3 Input

4.3.1 LIDAR-derived ponds and elevation

The Fort Stewart staff provided a LIDAR dataset of the installation developed by Earthdata in 2009. The GRASS LIDAR processing instructions described in the previous chapter were used to create a new vector map containing the outline of each of the LIDAR-based ponds; each was associated with the average depth of the pond, the maximum depth, the area of the pond, and the area of the watershed upstream from the pond. This information provided the primary information for the hydrologic component of the model.

4.3.2 Historic weather

Historic weather information was retrieved from the National Oceanic and Atmospheric Administration (NOAA) historic weather data archives for Fort Stewart, GA weather station (IDs 091544 and 093538) from 1950 through 2010. Missing data were filled in from nearby weather stations. These data provided the model with daily weather temperature means, minimums, maximums, and rainfall.

4.3.3 Ancillary maps for display

The LIDAR-based DEM was processed with r.shaded.relief to generate a shaded relief image used as a background to the model. Fort Stewart personnel supplied vector maps that showed roads, streams, and boundaries. These vector maps were also used for spatial orientation during inspection of the model.

4.4 Initialization

At model startup, weather and ponds are initialized. The model reads in weather data that provide daily rainfall and temperature for the simulation time. This is processed during read-in to create an internal table of weekly weather data. Ponds are initialized based on GIS-based pond vector files that are associated with variables including pond depth and size, and pond watershed size.

Finally, a graphical representation of the study areas is generated to allow visual tracking of pond depth along with graphs of temperature, population, and rainfall over time. Figure 7 shows a post-initialization visualization for an area interior to the installation. The background is a shaded relief image with the sun positioned due west of the LIDAR-based digital elevation. Darker blue lines outline the potential derived ephemeral ponds. For positioning, streams are overlaid in a lighter blue.

4.5 Submodels

4.5.1 Weather

The weather submodel identifies the weather data (minimum, average, and maximum temperature, total rainfall, and maximum daily rainfall) associated with the current week.

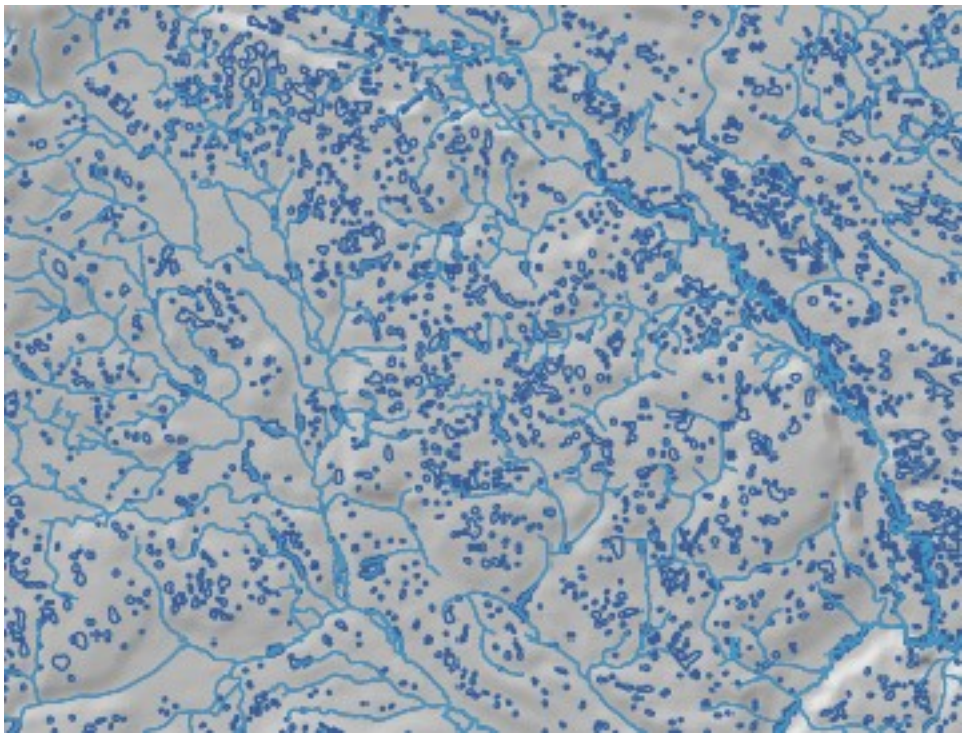


Figure 7. Typical model visualization after initialization.

4.5.2 Hydrology

Ponds are modeled as individual objects that take the shape of a cone. Each is initialized from a Shapefile that contains the pond vector outline, and pond depth, size, and watershed size. Most of the ephemeral ponds are generally round in shape and get larger as they deepen, suggesting that a geometric cone is an appropriate simplification. The maximum depth of the cone is the depth from the Shapefile and the diameter is based on the pond size, with the assumption that the pond is circular. The volume of the cone becomes the volume of the pond. With ponds established, a portion of the weekly rainfall can drain into the cone, and water in the cone can be lost at a given rate. The size of the pond, as might be observed by a visitor, is calculated from the cone shape and water volume.

The hydrology submodel uses the weekly weather data to update the water depth in each pond. For hydrology, ponds were modeled with a very simple lumped parameter approach using two parameters. Ponds each collect rainfall from their respective small watersheds and then lose water at a constant rate. A rainfall interception parameter establishes the amount of rainfall each week that does not flow into ponds. This takes into account rain that is intercepted by vegetation, water that is evaporated or transpired out of the soil, and water that is lost to groundwater.

The second parameter, pond drainage rate, is also a fixed value that sets the drop in water level from ponds each week due to evaporation, transpiration, and leakage. The first simulation experiment, described below, sets these two values for Fort Stewart as a whole. This assumes that these values are not significantly different across the installation due to differences in geology, soils, and vegetation. Future detailed geologic studies of the installation might allow each pond to be modeled with its own unique parameters.

In the resulting hydrology submodel, the weekly rainfall is reduced by the rainfall interception parameter. The resulting rainfall is multiplied by the size of each pond's watershed to establish the volume of water added to the pond. Each pond is modeled as an inverted cone with a representative fixed diameter and depth at its full point. The pond depth from the previous week is reduced by the pond drainage rate, establishing a new volume to which the additional new volume is added. The new pond depth and pond diameter is calculated and later used for establishing the survival of any eggs or larvae.

5 Hydrologic Calibration of Monitored Ephemeral Ponds

Fort Stewart personnel provided vector-based maps of ponds and spreadsheets of pond size history. Ponds are identified and given unique identifier codes, which allow cross-referencing between the mapped and tabulated data. Eleven of the ponds were visited in February 2009, and 83 ponds were visited December 2009 and April 2010, for a total of 185 observations. Visitors recorded the pond size subjectively as "full," "1/4 to 1/2 full," or "puddles." These observations were based on the apparent area of the pond with respect to the extent of each pond at its maximum size, but not pond depth.

The pond extent dataset described above was used to calibrate the two hydrologic parameters, rainfall interception rate, and pond drainage rate. The entire field-identified pond size information in the model was captured in tabular form. The model was run using weather data for the time period during which observations were made, with various combinations of rainfall interception and pond drainage rates. Pond and date combinations associated with field-collected information were captured in output files for post-model run analysis for each model run. The average squared difference between the field observations and the model calculations for pond size were calculated and tabulated for all pond data. (Table 1 lists the results of some of these calculations.) Calculations were done using 66 field observations of ponds identified as "full"; 23 observations of "1/4 to 1/2 full"; and eight observations of "puddles." These were given fill values of 0.8, 0.3, and 0.1 respectively. The squared differences calculations were weighted (1.0, 2.87, and 8.25 respectively) to allow equal overall weight for each fill level observation type.

$$\frac{8.25 * \sum_{f=.1}\left(c_f - o_f\right)^2 + 2.87 * \sum_{f=.3}\left(c_f - o_f\right)^2 + \sum_{f=.8}\left(c_f - o_f\right)^2}{8.25 * X + 2.87 * Y + Z} \qquad \text{Eq. 1}$$

where:

X = the number of observations where the observed depth is "puddles"
Y = the number of observations where the observed depth is "1/4 to 1/2 full"
Z = the number of observations where the observed depth is "full."

Table 1. Average squared difference between field and modeled pond size using all data.

		Rainfall Intercept (in.)/week							
		1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6
Drainage rate per week in meters	0.01	0.352	0.330	0.304	0.276	0.233	0.198	0.156	0.138
	0.02	0.243	0.219	0.146	0.182	0.114	0.091	0.062	0.064
	0.03	0.172	0.144	0.117	0.096	0.081	0.074	0.070	0.081
	0.04	0.124	0.105	0.089	0.081	0.079	0.081	0.097	0.109
	0.05	0.102	0.090	0.083	0.084	0.088	0.089	0.124	0.135

Table 1 lists the average squared difference between calculated and observed pond depths for various combinations of rainfall intercept and pond drainage rates. A 3rd degree polynomial applied across these data yields a best fit for the two hydrologic parameters of 2.5 in. per week for rainfall interception and 0.025m per week for pond drainage.

This calibration is based on several assumptions:

1. The pond 3-D shape is appropriately modeled as a cone. While the actual shape is perhaps more accurately described as a bowl, the simplified cone serves well.
2. Pond depth and size can be modeled with the two-parameter approach (rain intercept and weekly drainage rate). Unless the landscape is impermeable, not all rainfall flows off the watershed. Rainfall can be held in the leaves of vegetation where it can evaporate over time. Rain that hits the ground can be absorbed by the soil, conveyed to groundwater areas, and then transpired over time by local vegetation. The rain intercept rate attempts to capture the combined effect of all of these processes in one parameter. Once rainfall accumulates in a pond, that water can similarly evaporate, be transpired by vegetation, or make its way into groundwater. The weekly drainage rate captures all of these processes in one value.
3. All ponds can be calibrated with the same values. This is likely the most challenging assumption, which relies on the notion that the formation of these ponds involved similar processes — over time, over the same substrate, and involving the same soil types. Soil maps are available for the area, but differences between ponds might still require different calibrations. It is anticipated that more detailed pond depth measurements will allow each pond to be calibrated separately.

6 Conclusions

Many imperiled and at-risk species on military installations occupy very narrow or specific niches on the landscape. Ephemeral ponds are one such niche that provide amphibians with breeding areas that are largely predator-free. Giving land managers the ability to locate ephemeral ponds will help them to better manage imperiled and at-risk populations. This work developed a modeling approach to quickly and accurately locate and measure ephemeral ponds using the Geographic Resources Analysis Support System Geographic Information System (GRASS GIS) software, and then to add behavior to those ponds within the NetLogo spatially explicit simulation-modeling environment.

Results show that LIDAR technology, which reflects laser flashes off surfaces to capture a 3-D structure of the ground, vegetation from ground cover to trees, and manmade structures, provides a powerful tool for rapidly acquiring information about landscapes that can lead to near real-time understanding of the state of landscapes. This work also described and demonstrated procedures that select LIDAR ground hits and process those data into DEMs to accurately locate and measure ephemeral ponds. A NetLogo-based model developed as part of this research provides this capability for Fort Stewart, GA.

It is concluded that this identification of the structure of ephemeral ponds and the extent of the watersheds that feed them will make it possible to predict the historic depths of these ponds and to predict their ability to support breeding of specific species.

Acronyms and Abbreviations

Term	Definition
2-D	Two Dimensional
3-D	Three Dimensional
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
CEERD	US Army Corps of Engineers, Engineer Research and Development Center
CERL	Construction Engineering Research Laboratory
DC	District of Columbia
DEM	Digital Elevation Model
ERDC	Engineer Research and Development Center
FAQ	Frequently Asked Questions
GIS	Geographic Information System
GRASS	Geographic Resources Analysis Support System
ID	Identification
IDW	Inverse Distance Weighting
LIDAR	Light Detection and Ranging
NOAA	National Oceanic and Atmospheric Administration
NSN	National Supply Number
OMB	Office of Management and Budget
OS	Operating System
RST	Regularized Spline with Tension
SAR	Same As Report
SF	Standard Form
TR	Technical Report
UTM	Universal Transverse Mercator
WWW	World Wide Web

References

GRASS Development Team. 2012. Welcome to GRASS GIS. Website. Accessed 27 June 2012, http://grass.osgeo.org/

Neteler, M. and H. Mitasova. 2007. Open source GIS: A GRASS GIS Approach. New York: Springer.

Railsback, S. F., and V. Grimm. 2011. Agent-based and individual-based modeling: A practical introduction. Princeton, NJ: Princeton University Press.

US Fish and Wildlife Service (USFWS). 1973. Endangered Species Act of 1973, as Amended through the 108th Congress. PL 93-205; 16 US Code 1531 et seq., as amended. Website, http://www.fws.gov/endangered/pdfs/ESAall.pdf

Westervelt, J. D., and G. L. Cohen (eds.). 2012. Ecologist-developed spatially-explicit dynamic landscape models (modeling dynamic systems). New York: Springer.

Westervelt, J. D., M. Shapiro, W. D. Goran, and D. P. Gerdes. 1992. Geographic Resources Analysis Support System (GRASS) Version 4.0 User's Reference Manual. N-87/22. Champaign, IL" Construction Engineering Research Laboratory (CERL).

Wilensky, U. 1999. NetLogo. Computer software. Evanston, IL: Northwestern University Center for Connected Learning and Computer-Based Modeling, http://ccl.northwestern.edu/netlogo/

Appendix A: Sample Scripts

```
!/bin/sh

# indicate the number of simultaneous processes allowed
processors=10

north=3555200
south=3523700
west=415300
east=471550

size=6300
ew=`expr $east - $west`
ns=`expr $north - $south`

#### Create the point file for each sub area

p=0
for col in 0 1 2 3 4 5 6 7 8
do
  for row in 0 1 2 3 4
  do
    w=`expr $col \* $size`
    w=`expr $w + $west`
    e=`expr $w + $size`
    s=`expr $row \* $size`
    s=`expr $s + $south`
    n=`expr $s + $size`
    # adjust to ensure patch overlap
    n=`expr $n + 25`
    s=`expr $s - 25`
    e=`expr $e + 25`
    w=`expr $w - 25`

    echo n:$n s:$s w:$w e:$e res=5

    g.region w=$w e=$e n=$n s=$s res=5

    tmp=`echo ${row}_${col}`

    v.in.ascii -z -r -b all+10.xyz out=lidar_$tmp format=point
fs=, z=3 --o &

    p=`expr $p + 1`
    if [ $p -ge $processors ]
    then
      wait
      p=0
    fi
  done
done
wait
```

```
#### Build the elevation, aspect, and slope maps

p=0
for col in 0 1 2 3 4 5 6 7 8
do
  for row in 0 1 2 3 4
  do
    tmp=`echo ${row}_${col}`
    (
      #### Build the vector topology
      g.region vect=lidar_$tmp
      v.build lidar_$tmp --q
      #### Create the first DEM for each area,
      #### along with profile curvature
      g.region vect=lidar_$tmp
      v.surf.rst lidar_$tmp elev=lidar_elev_$tmp layer=0 \
        pcurv=lidar_pcurv_$tmp --o --q
      #### Mask out the rasters in the DEM with a positive
      #### (convex) curvature;  assume these are associated
      #### with vegetation
      g.region rast=lidar_elev_$tmp
      r.mapcalc lidar_elev_$tmp=\
        if\(lidar_pcurv_$tmp \< 0, lidar_elev_$tmp, null\(\)\)
      #### Convert the remaining rasters to new point files
      g.region rast=lidar_elev_$tmp
      r.to.vect -z feature=point in=lidar_elev_$tmp \
        out=lidar_$tmp --o --q
      #### Run v.surf.rst a second time using the
      #### just-created point files
      g.region rast=lidar_elev_$tmp
      v.surf.rst layer=0 in=lidar_$tmp layer=0 \
        elev=lidar_elev_$tmp \
        slope=lidar_slope_$tmp \
        aspect=lidar_aspect_$tmp --o  --q
    ) &
    ## Check to see if the max # of processes is running.
    ##  If so, wait for completion before continuing
    p=`expr $p + 1`
    if [ $p -ge $processors ]
    then
      wait
      p=0
    fi
  done
done
wait

#### Trim the edges off the final maps

for col in 0 1 2 3 4 5 6 7 8
do
  for row in 0 1 2 3 4
  do
    tmp=`echo ${row}_${col}`
```

```
    g.region rast=lidar_elev_$tmp
    g.region -g > /tmp/region
    source /tmp/region
    n=`expr $n - 15`
    s=`expr $s + 15`
    e=`expr $e - 15`
    w=`expr $w + 15`
    g.region -p
    g.region n=$n s=$s e=$e w=$w
    g.region -p
    for m in lidar_elev_$tmp lidar_elev_$tmp lidar_slope_$tmp \
      lidar_aspect_$tmp
    do
      r.mapcalc ${m}=${m} &
    done
    wait
  done
done

#### Combine each set of final maps into one complete map

g.region w=$west e=$east n=$north s=$south res=5
for m in lidar_elev lidar_elev lidar_slope lidar_aspect
do
  r.mapcalc "$m=\
    if(! isnull(${m}_0_0), ${m}_0_0, \
    if(! isnull(${m}_0_1), ${m}_0_1, \
    if(! isnull(${m}_0_2), ${m}_0_2, \
    if(! isnull(${m}_0_3), ${m}_0_3, \
    if(! isnull(${m}_0_4), ${m}_0_4, \
    if(! isnull(${m}_0_5), ${m}_0_5, \
    if(! isnull(${m}_0_6), ${m}_0_6, \
    if(! isnull(${m}_0_7), ${m}_0_7, \
    if(! isnull(${m}_0_8), ${m}_0_8, \
    if(! isnull(${m}_1_0), ${m}_1_0, \
    if(! isnull(${m}_1_1), ${m}_1_1, \
    if(! isnull(${m}_1_2), ${m}_1_2, \
    if(! isnull(${m}_1_3), ${m}_1_3, \
    if(! isnull(${m}_1_4), ${m}_1_4, \
    if(! isnull(${m}_1_5), ${m}_1_5, \
    if(! isnull(${m}_1_6), ${m}_1_6, \
    if(! isnull(${m}_1_7), ${m}_1_7, \
    if(! isnull(${m}_1_8), ${m}_1_8, \
    if(! isnull(${m}_2_0), ${m}_2_0, \
    if(! isnull(${m}_2_1), ${m}_2_1, \
    if(! isnull(${m}_2_2), ${m}_2_2, \
    if(! isnull(${m}_2_3), ${m}_2_3, \
    if(! isnull(${m}_2_4), ${m}_2_4, \
    if(! isnull(${m}_2_5), ${m}_2_5, \
    if(! isnull(${m}_2_6), ${m}_2_6, \
    if(! isnull(${m}_2_7), ${m}_2_7, \
    if(! isnull(${m}_2_8), ${m}_2_8, \
    null()))))))))))))))))))))))))))) "
done
```

Appendix B: Model

```
extensions [ gis ]
__includes ["weather.nls"]

breed [ ponds pond ]

globals
[
  dataset
  streams-dataset
  pond-lidar-dataset
  T-max                  ; these three "T-" lists hold temp data
  T-min
  T-mean
  year-list              ; list of year associated with last day
of each week
  year
  month-list             ; list of month associated with last day
of each week
  month
  last-month
  day-list               ; list of day associated with last day
of each week
  day
  week
  temperature            ; the temperature for the current week
  max-daily-rainfall     ; the list holding daily rainfall max
  max-1-day-rainfall     ; the max-rainfall variable for
  weekly-rainfall        ; the list holding rainfall totals
  weeks-rainfall         ; the rainfall for the current week
  report-file            ; file that will contain output
]

patches-own
[ shaded-color           ; gis map input
  shaded-color2          ; temp variable for patch color
  tmp                    ; general purpose temp variable
]

ponds-own
[ pond-ID       ; Pond category ID read from the pond vector map
  watershed-m   ; size of the pond's watershed in square meters
  pond-area-m   ; size of the pond in square meters
  avg-depth     ; average depth of the pond
  max-depth     ; max depth of the pond re
  max-radius    ; maximum radius of the pond when full
  max-volume    ; maximum volume of the pond
  tan-apex      ; intermediate calculation of pond depth
  pond-vol      ; current volume of the pond
  water-depth   ; current depth of the pond
  water-radius  ; This is the radius of water in the pond
```

```
  pond-outline  ; GIS vector feature containing pond vertices
]

to initialize
  clear-all
  initialize-weather
  initialize-landscape
  ;do-plots
  setup-pond-reporting
  reset-ticks
end

to go
  if ticks = weeks-weather-data-to-read-in [stop]
  update-weather
  update-landscape
  draw-ponds
  tick
end

to reset
  clear-all-plots
  reset-ticks
end

to setup-pond-reporting
  set report-file (word "results/report-full-" et-leak-rate "-"
    rain-absorption-rate .”txt")
  if (file-exists? report-file) [ file-delete report-file ]
end

to create-report
  set report-file (word "results/report-full-" et-leak-rate "-"
    rain-absorption-rate "-" rain-adjust .”txt")
  if (file-exists? report-file) [ file-delete report-file ]

  file-open report-file

  ask ponds
  [
    file-type pond-id
    file-type " "
  ]
  file-close
end

to initialize-landscape
  initialize-maps
  initialize-ponds
end

to initialize-maps
  let map-base .”./maps/full"
  ; Establish coordinate transformation between GIS maps
  ; and Netlogo patches

  set dataset gis:load-dataset (word map-base
```

```
    "/rast/shadedmap.asc")

  resize-world 0 (gis:width-of dataset) 0 (gis:height-of dataset)
  set-patch-size ( 900 / gis:width-of dataset )
  ; Establish coordinate transformation internal to gis extension
  gis:set-transformation
    (gis:envelope-of dataset)
    (list min-pxcor max-pxcor min-pycor max-pycor)

  gis:apply-raster dataset shaded-color

  print "Reading maps"
  ; Read in pond (vector) map
  set pond-lidar-dataset gis:load-dataset
    (word map-base  "/vect/ponds_lidar/ponds_lidar.shp")
  set streams-dataset     gis:load-dataset
    (word map-base  "/vect/streams/streams.shp")

  display-background
  display-vector-maps
end

to initialize-ponds
  ; Create ponds
  print (word "Attempting to create "
    length gis:feature-list-of pond-lidar-dataset " ponds")

  ; Loop through the lidar ponds
  foreach gis:feature-list-of pond-lidar-dataset
  [ create-ponds 1
    [
      ;Store the pond ID
      set pond-ID gis:property-value ? "CAT"
      if pond-id = nobody [ die ]

      ; Save the pond outline for later display
      set pond-outline ?

      ; capture hydrology info
      set watershed-m (gis:property-value ? "ACCUM_HA") * 10000
; convert from ha to m^2
      set pond-area-m (gis:property-value ? "AREA_HA") * 10000

      ; throw out ponds that have huge watersheds
      if (watershed-m / pond-area-m) >= max-watershed-pond-size-
ratio [ die ]

      ; Set pond geometry based on an inverted cone
      set avg-depth   gis:property-value ? "AVG_M"
      set max-depth   gis:property-value ? "MAX_M"
      set max-radius ( sqrt ( pond-area-m / pi ) )
      set tan-apex max-radius / max-depth
      if tan-apex = 0 [ die ]

      ; Set the pond as empty to start
      ; maximum volume of water (m^3) that pond will hold.
      ; Pond geometry is conical.
```

```
      set max-volume (1 / 3) * pi * (max-radius ^ 2) * max-depth
      ; volume of water (m^3) initialized in pond
      set pond-vol 0
      ; surface area radius of water currently in pond
      set water-radius ( pond-vol * 3 * tan-apex / pi ) ^ (1 / 3)
      set water-depth ( water-radius / tan-apex )
    ]
  ]

  print (word "Created " count ponds " ponds")
end

to update-landscape
  ask ponds
  [
    ; Drain the pond
    set water-depth water-depth - et-leak-rate
    ; Keep the pond depth > 0
    if water-depth < 0 [ set water-depth 0 ]
    set water-radius water-depth * tan-apex
    set pond-vol (water-radius ^ 3) / ( 3 * tan-apex / pi )

    ; Fill the pond
    ; Convert rainfall (week's-rainfall in inches, so convert
    ;   to meters) to volume of water added to watershed
    let watershed-volume weeks-rainfall * 0.0254 * watershed-m
    ; Add that volume to pond
    set pond-vol pond-vol + watershed-volume
    ; Cap the pond growth to its max possible size
    if pond-vol > max-volume [ set pond-vol max-volume ]
    ; Translate new pond volume to depth and diameter
    ;   (see code in setup)
    set water-radius ( pond-vol * 3 * tan-apex / pi ) ^ ( 1 / 3 )
    set water-depth ( water-radius / tan-apex )
  ]
end

to draw-ponds
  ask ponds
  [
    ; Set the color based on depth (darker blue will be deeper)
    gis:set-drawing-color 90 + 9 * ((5 - water-depth) / 5)
    gis:draw pond-outline .5
  ]
end

to display-vector-maps
  gis:set-drawing-color 4       gis:draw streams-dataset 1
end

to display-background
  ask patches [ set tmp 255 - shaded-color ]
  let max-val [ tmp ] of max-one-of patches [ tmp ]
  let min-val [ tmp ] of min-one-of patches [ tmp ]
  let range max-val - min-val
  ask patches
  [
```

```
    set shaded-color2 9.9 - ( 9.9 * (tmp - min-val) / range )
    set pcolor shaded-color2
  ]
end



;;; WEATHER DATA READ-IN AND DELIVERY
to initialize-weather
  set year-list []           ; list of year associated with last
                             ;   day of each week
  set month-list []          ; list of month associated with last
                             ;   day of each week
  set day-list []            ; list of day associated with last
                             ;   day of each week
  set T-max []               ; list of max weekly T (F)
  set T-min []               ; list of min weekly T (F)
  set T-mean []              ; list of mean weekly T (F)
  set weekly-rainfall []     ; daily rainfall (inches) grouped
                             ;   into 7-day sets
  set max-daily-rainfall []  ; list of max 1-day rainfall for
                             ;   each week
  file-close
  file-open "weather.csv"
  let next-line file-read-line
  type "Reading in weather data:  " print next-line

  ; create temporary lists for assembling weekly weather data
  let Tmax-temp 0
  let Tmin-temp 0
  let Tmean-temp []
  let rain-this-week []
  let rain-max-temp 0
  let skip-this-item 0
  let x 0

  let read-in weeks-weather-data-to-read-in
  while [ read-in > 0 ]
    [ set Tmax-temp 0
      set Tmin-temp 0
      set Tmean-temp []
      set rain-this-week []
      set rain-max-temp 0
      ; ignore weather station ID# and date
      repeat 5 [ set skip-this-item file-read ]
      set Tmax-temp file-read
      set Tmin-temp file-read
      set Tmean-temp lput file-read Tmean-temp
      repeat 2 [ set skip-this-item file-read ]
      set rain-this-week lput file-read rain-this-week

      repeat 6
      [ repeat 2 [ set skip-this-item file-read ]
        set year file-read
        set month file-read
        set day file-read
```

```
        set x file-read
        if x > Tmax-temp [ set Tmax-temp x ]
        set x file-read
        ; store the lowest temp this week
        if x < Tmin-temp [ set Tmin-temp x ]
        set x file-read
        set Tmean-temp lput x Tmean-temp
        repeat 2 [ set skip-this-item file-read ]
        set x file-read
        set rain-this-week lput x rain-this-week
        if x > rain-max-temp [ set rain-max-temp x ]
      ]

    set year-list  lput year year-list
    set month-list lput month month-list
    set day-list   lput day day-list
    set T-max lput Tmax-temp T-max
    set T-min lput Tmin-temp T-min
    set T-mean lput ( mean Tmean-temp ) T-mean
    set max-daily-rainfall lput rain-max-temp max-daily-rainfall
    while [ length rain-this-week < 7 ]
    [
      set rain-this-week lput mean rain-this-week rain-this-week
    ]
    ; sum the week's rainfall and add the total to the
    ; list of weekly rainfalls
    set weekly-rainfall lput
      ( reduce [ ?1 + ?2 ] rain-this-week ) weekly-rainfall

    set read-in ( read-in - 1 )
  ]
  file-close
end


to update-weather
  ; select current temperature and rainfall from weather lists

  set year        item ticks year-list
  set last-month  month
  set month       item ticks month-list
  set day         item ticks day-list
  ifelse (month = 1 AND day <= 7)
    [ set week 0 ]
    [ set week week + 1 ]
  set temperature item ticks T-mean
  set weeks-rainfall ( item ticks weekly-rainfall ) *
    ((100 + rain-adjust) / 100 ) - rain-absorption-rate
  if weeks-rainfall < 0
    [ set weeks-rainfall 0 ]
  set max-1-day-rainfall
    (item ticks max-daily-rainfall) * ((100 + rain-adjust) / 100)
end
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
26-09-2012	Final	

4. TITLE AND SUBTITLE
Digital Discovery of Ephemeral Ponds

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT

6. AUTHOR(S)
James D. Westervelt

5d. PROJECT NUMBER
IDIR

5e. TASK NUMBER

5f. WORK UNIT NUMBER
33143

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Engineer Research and Development Center (ERDC)
Construction Engineering Research Laboratory (CERL)
PO Box 9005,
Champaign, IL 61826-9005

8. PERFORMING ORGANIZATION REPORT NUMBER
ERDC/CERL TR-12-21

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Engineer Research and Development Center (ERDC)
Environmental Laboratory (EL)
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

10. SPONSOR/MONITOR'S ACRONYM(S)
CEERD-EM-D

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEME
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The US Endangered Species Act requires Federal agency land owners, including military installations, to manage their lands in a manner that enhances the survival of Federally listed species. Many species become at risk due to the loss of "ephemeral ponds," depressions in the landscape that occasionally become ponds after sufficient rainfall. This report developed a modeling approach using the Geographic Resources Analysis Support System Geographic Information System (GRASS GIS) software, augmented with a NetLogo-based model to add behavior to those ponds within the NetLogo spatially explicit simulation-modeling environment. This tool will allow installation land managers to quickly and accurately locate and measure ephemeral ponds to support species that rely on that environment for breeding.

15. SUBJECT TERMS
GRASS, GIS, land management, simulation modeling

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)
Unclassified	Unclassified	Unclassified	SAR	40	